# Nonvolatile System Tables

*Commentary on nonvolatile memory usage*

Mon, Oct 23, 2000

IRM system software requires a set of nonvolatile tables of configuration information, especially that relating to accelerator devices. This note lists the tables in use, describing something about each. The tables are allocated from 0.5 MB of nonvolatile memory, which is included on the MVME-162 board. The new MVME-2401 PowerPC-based boards include 2.0 MB of nonvolatile memory. The first 0.5 MB is used for nonvolatile tables; the remaining 1.5 MB is used for a memory file system.

***System table directory***

A table directory is allocated in the first 256 bytes of nonvolatile memory. It has room for 32 eight-byte entries using the following format:

| *Field* | *Size* | *Meaning* |
|---|---|---|
| `numE` | 2 | Number of entries |
| `eSiz` | 2 | Size of each entry |
| `tPtr` | 4 | Base address of table |

Each table is thus considered a simple array of structured entries with a fixed size for each entry. A table number is the corresponding entry number of the table directory entry. The table directory is assumed to be located at the start of the nonvolatile memory; it is not itself represented by a table directory entry. At system reset time, the table directory's contents are checksummed against the reference checksum recorded in the last entry of the table directory. (The last entry is also used as the download area for local/page applications. To gain space for the checksum word, the base address for this entry is assumed to start on a 64K-byte boundary; i.e., its base address low word is `0x0000`.)

***Analog channel tables***

This set of tables provides support for analog channels. By default, they are allocated for 1024 channels, but this can be increased to 2048 channels without altering the allocation of nonvolatile memory. The analog tables are:

| *Name* | *#* | *`eSiz`* | *Purpose* |
|---|---|---|---|
| `ADATA` | 0 | 16 | Analog reading, setting, alarm data/flags, etc. |
| `ADESC` | 1 | 64 | Analog descriptor: name, text, scaling control info, etc. |
| `CINFO` | 25 | 8 | Variable-size entries holding sparse extra analog info |
| `FDATA` | 26 | 16 | Floating point reading, setting, nominal, tolerance |
| `ADEVX` | 23 | 4 | Acnet device index values |

All of the analog tables are indexed by analog channel number, except for `CINFO`, which uses entries that include the channel number for specifying extra information about a relatively few channels. Each entry also specifies a type code and its own size, which must be a multiple of 8 bytes. A common use for this is to relate fast digitizer parameters to those few channels for which it is relevant.

***Binary bit/byte tables***

This set of tables supports access to binary (digital) information.

| *Name* | *#* | *`eSiz`* | *Purpose* |
|---|---|---|---|
| `BALRM` | 2 | 4 | Binary alarm flags, alarm count |
| `BDESC` | 3 | 16 | Text field for each binary bit |
| `BBYTE` | 5 | 1 | Binary status byte readings |
| `BADDR` | 11 | 4 | Binary addresses used to refer to binary data bytes |
| `CSTAT` | 24 | 32 | Construct digital data to be stored as analog "reading" |

The `BALRM` and `BDESC` tables are indexed by binary bit number. The `BBYTE` and `BADDR` tables are indexed by binary byte number. Each binary byte holds 8 binary bits, as usual. For example, byte 7 includes Bits 63–56, in order of most significant to least significant bit. The `CSTAT` table entries refer to byte numbers whose `BBYTE` data values are to be sampled, masked shifted, and OR'd to produce a binary status word that is accessed as an analog channel. Those channels being used to hold binary status words are indicated via a bit set in the alarm flags word for that channel (mask=`0x4000`). During the analog alarm scan, such channels are checked for alarms using digital operations masking and XOR'ing rather than the usual range-checking.

***Page/local application tables***

| *Name* | *#* | *eSiz* | *Purpose* |
|---|---|---|---|
| `PAGEP` | 6 | 20 | Page 4-character name (`PAGExxxx`), 16-character title |
| `PAGEM` | 7 | 128 | Page-private memory, auto-page parameters |
| `CODES` | 9 | 32 | Directory of program "files" in memory file system. |
| `LATBL` | 14 | 32 | Local application name, private memory ptr, params |

The `PAGEP` and `PAGEM` tables are indexed by page number, which range from 0 to 31. The page #0 entry for `PAGEM` has special system-wide uses, including holding the target multicast node number for requests, the target node number for Classic alarms, the nonvolatile copy of the date/time, serial port params, and the motor step period. The `PAGEP` holds a 4-character name that is a key into the `CODES` table entry that holds the pointer to the file itself. The `LATBL` has as many entries as needed for specifying all the local application instances required.

***Network tables***

Several tables relate to the network support used by IRMs:

| *Name* | *#* | *eSiz* | *Purpose* |
|---|---|---|---|
| `TRING` | 30 | 1024 | Many general network tables included |
| `IPARP` | 28 | 16 | Relates IP/physical addresses, active port#s |
| `IPNAT` | 27 | 8 | Relates node numbers with IP addresses via DNS |
| `TRUNK` | 20 | 1024 | Holds up to 256 IP addresses per Acnet "trunk" |
| `OUTPQ` | 12 | 8 | Network output message pointer queue |

Although `TRING` got its name from token ring, the first supported network, it also has many small network-related tables that may be used by ethernet or arcnet. It includes a "Transmit parameter list" structure that supports the queuing of multiple frames/datagrams for network transmission. The `IPARP` table serves as a typical ARP table, but it also keeps track of the active UDP port numbers in use for each IP address. This supports referring to a socket (IP address plus UDP port#) by a 16-bit "pseudo node number." The `IPNAT` table holds the IP address for each node number. The IP addresses are obtained from the local Domain Name Server via the help of the `DNSQ` local application. Entries are placed into this table automatically when an attempt is made by the local node to target a node number for the first time ever.

The `TRUNK` table serves to relate Acnet node numbers to IP addresses. In a sense, it is similar to `IPNAT`, but it is downloaded from an Acnet operational node, as Acnet does not use the DNS.

A pointer to each message block containing a message that is prepared to be sent to the network is first placed into `OUTPQ`. When this queue is flushed, successive entries that refer to the same destination node can be combined within a single datagram. This works to advantage for the Classic and Acnet protocols.

***D0 protocol tables***

A few tables were allocated for use in support of D0-specific needs in Run I.

| *Name* | *#* | *eSiz* | *Purpose* |
|---|---|---|---|
| `AADIB` | 21 | 32 | Analog device info block |
| `BADIB` | 22 | 32 | Binary device info block |
| `CADIB` | 23 | 32 | Comment device info block |
| `MMAPS` | 16 | 8 | Memory-maps for downloaded data |

The first three tables are used to support D0 protocol alarms reporting. The tables were downloaded by D0 operations as part of system configuration, and the info was used to build alarm messages. The `MMAPS` table provided a way to download data to D0-designed electronics in which some fields had to be skipped to avoid bus errors or read-only registers. These tables have use only for D0, and since D0 is being upgraded to an EPICS control system, they can be released for other purposes. The tentative purposes are as follows:

| *Name* | *#* | *eSiz* | *Purpose* |
|---|---|---|---|
| ----- | 21 | 0 | spare |
| ----- | 22 | 0 | spare |
| ADEVX | 23 | 4 | Acnet device indexes |
| MMAPS | 16 | 8 | Memory-maps for downloaded data. (?) |

The ADEVX table allows storage of Acnet central database device indexes to facilitate alarms reporting to Acnet. The old Acnet scheme of using a front-end-specific EMC is to be retired. The new ADEVX table can be considered another analog channel table. The MMAPS table may be replaced in the future.

***Miscellaneous tables***

These tables have various odd functions:

| *Name* | *#* | *eSiz* | *Purpose* |
|---|---|---|---|
| RDATA | 4 | 16 | The Data Access Table updates the data pool. |
| LISTP | 8 | 8 | Entries manage message-ids used in requests |
| PRNTQ | 13 | 4 | Holds ptrs to text blocks awaiting serial output |
| SERIQ | 19 | 1 | Buffers serial input characters. |
| CPROQ | 16 | 16 | Pointers to co-processor message queues |
| Q1553 | 17 | 4 | Pointers to 1553 memory command queues |
| DSTRM | 18 | 32 | Definition of data stream queues available |
| MAP32 | 29 | 2 | High words of addresses used in analog control fields |

The RDATA table is processed every cycle to update the data pool, which is mostly the contents of the ADATA and BBYTE tables. All enabled local applications are also called as part of this process, and they may also update some of the analog and/or binary data.

When multiple requests are made under the Classic protocol, the LISTP table organizes the use of message-ids that serve to distinguish replies arriving from various active data requests.

Serial port output is queued via entries in the PRNTQ. Serial input characters are queued in the SERIQ table for subsequent processing by a page or local application.

The CPROQ table allows for simple message communication with other processors that share the same VME memory bus. Analog control can be passed as messages through such a queue, for example.

The Q1553 table merely keeps pointers to the active 1553 command queues that reside on the 1553 controller boards. This was used heavily for D0 Run I nodes, but it is needed for any node that uses the 1553 field bus communications.

The DSTRM supports access to data streams, which are a formalized scheme for the use of circular buffers. An example of their use is for network diagnostics, or even for task timing diagnostics. Heavy writing access to such queues can cause them to wrap frequently, so that a client may need to empty them often to avoid losing information.

The MAP32 table was implemented for deriving 32-bit memory addresses from a 24-bit representation. It is needed for many types of analog control fields in the analog channel descriptor structure, in which a memory address must be specified in 24 bits. The upper 8 bits of the 24 bits index into this table to obtain the upper 16 bits of the target address. The low 16 bits are used directly. It's a kind of memory map scheme. Since this table is especially needed to interpret analog control fields, it can be considered another analog channel table. It may be usable in the future for BADDR entries as well. An advantage to doing so would be that the upper 16 bits of addresses would not have to appear so many places in system tables.